- 数据库技术是计算机应用技术中最重要的组成部分，许多应用程序都离不开数据库的存取操作。VB.NET自身不具备对数据库进行操作的功能，它对数据库的处理是通过.NET FrameWork SDK中面向数据库编程的类库实现的。

9.1 ADO.NET概述

- **9.1.1 认识ADO.NET对象**
- **ADO.NET是由微软Microsoft ActiveX Data Object(ADO)升级发展而来的，是.NET平台全新的数据库访问技术，提供了.NET数据库应用程序的编程接口。它可以被看作是一个介于数据源和数据使用者之间的转换器，能够让开发人员更加方便的在应用程序中使用和操作数据。在ADO.NET中，大量复杂的数据操作的代码被封装起来，所以用户只需编写少量的代码即可完成大量的数据库操作。**

- **1. ADO.NET的名称空间**
- **ADO.NET是围绕System.Data 基本名称空间设计，其他名称空间都是从System.Data派生出来的。在使用中，如果要引用名称空间中的类，必须要导入相应的名称空间。导入名称空间的基本语法如下：**
- **Imports namespace**
- **例如，要使用OleDb前缀的类，则必须导入System.Data.OleDb名称空间。具体的语法如下：**
- **Imports System.Data.OleDb**

- **2. ADO.NET数据提供者**
- **在一个典型的.NET数据库应用程序中，用户通过一个.NET的数据提供者同数据库交互。**
- **ADO.NET支持两种数据提供者：SQL Server.NET数据提供者和OLE DB.NET数据提供者。**

类库	名称空间	访问数据库
System.Data.SQL	System.Data.sqlClient	SQL Server
System.Data.ADO	System.Data.OleDb	OLE DB，如Access

- **3. ADO.NET的工作过程**
- **ADO.NET对数据的操作基本上需要四个步骤：**
- **A. 使用Connection对象建立与数据源的连接。**
- **B. 使用Command对象给出对数据操作的命令，如查询或者更新数据等。**
- **C. 执行命令对象并显示数据结果。**
- **D. 最后关闭数据库连接。**

9.1.2 ADO.NET的常用对象

表 9-1　ADO.NET 的常用对象

对　象	说　明
Connection 对象	用于应用程序与数据源的连接
Command 对象	允许对数据库进行操作，如数据添加、删除、修改等。
DataAdapter 对象	数据适配器，用于对 OLEDB 数据源的读写，并可以填充 DataSet 和更新数据源
DataReader 对象	对数据库进行读取，并且时刻与服务器保持连接
DataSet 数据集	保存了在 ADO.NET 应用程序中的内存数据值

- 其中，Command 对象和DataReader对象可以进行查询，检查查询结果，用Command 对象还可进行更新、插入、删除操作，进行动态查询和调用存储过程。
- DataAdapter 对象用于脱机处理数据，可以使用DataAdapter对象 的Fill 方法将数据从数据库取出，放入内存中的DataSet数据集。DataAdapter对象可以对内存中的DataSet数据集进行更新、插入、删除操作，最后用Update 方法将对DataSet数据集的更新、插入、删除操作的结果保存到数据库中。

9.2　连接和操作数据库

- **9.2.1 使用Connection对象连接数据库**
- **数据库连接通常指的是应用程序和数据库的连接。**
- **以连接Access数据库为例。**
- **1、首先要创建数据库（略）**
- **2、连接Access 数据库**

- 用Connection对象连接Access 2010数据库。步骤如下：
- 1）设置数据库连接字符串
- Access数据库的连接字串示例代码如下：
- Dim strconn as String
- strconn=” Provider=Microsoft.ACE.OLEDB.12.0; Data Source='d:\data\schedules.accdb '”
- 注意：由于不同的Access版本间存在一些差异，所以要注意相应的OLE DB的版本问题。
- 2）创建指定ConnectionString属性的Connection对象
- 在声明了数据库连接字串后，就可以实例化一个OleDbConnection对象进行连接，示例代码如下：
- Dim objconn as OleDbConnection=New OleDbConnection(strconn)

3）调用Connection对象的Open()方法打开连接并测试

可以调用Open()方法来打开连接。也可以使用Close()方法来关闭连接，示例代码如下：

```
 objconn.Open()                    ‘打开连接
if(objconn.state= ConnectionState.Open)
      MessageBox.Show("连接成功！")  ‘提示连接成功
      objconnClose()
else
      MessageBox.Show("连接失败，请检查相关参数！")  ‘提示连接失败
End if
```

- **9.2.2 Command对象的使用**
- **使用Command对象有几个重要的步骤：**
- **1）创建Command对象。**
- **2）指定对象数据库连接和该对象关联的SQL命令。**
- **3）调用对象的Execute方法。**

- **1. 创建Command对象**
- **创建一个支持OLE DB 数据源的Command对象的语法结构如下：**
- **Dim *Command对象*as OleDbCommand=new OleDbCommand (String, OleDbConnection)**
- **为了能够操纵数据库中的数据，用户可以建立一个OleDbCommand对象,并含有具有合适参数的SQL表达式和连接，示例代码如下：**
- **Dim objCmd as New OleDbCommand（”Select * from Instructor”,objConn）**
- **也可以写成：**
- **Dim objCmd as OleDbCommand=New OleDbCommand()**
- **objCmd.Connection=objConn**
- **objCmd.CommandText=”Select * from Instructor”**

2. Command对象的属性

- 指定对象数据库连接和该对象关联的SQL命令，需要通过Command对象的属性实现。

表 9-6　Command 对象的主要属性

属　　性	说　　明
Connection	获取或设置 Command 的数据库连接对象
CommandText	指定数据库的查询信息
CommandType	指定数据查询信息的类型
Parameters	获取或设置 Command 对象的参数集合

3. Command对象的方法

Command对象执行SQL命令有三种方法。

表 9-7　Command 对象的方法

方　法	说　　明
ExecuteReader	将 CommandText 发送到 Connection 并生成一个 DataReader 对象
ExecuteScalar	执行查询，并返回查询所返回的结果集中的第一行第一列，忽略其他行或列
ExecuteNonQuery	针对 Connection 执行 SQL 语句并返回受影响的行数

- 例9.1 检索教师信息表“Instructor”中的职工号为“1001”的教师的信息，并将结果显示。（见教材第220页）
- 例9.2 统计课程信息表“Course”中的课程数量，并将结果显示。（见教材第221页）
- 例9.3 从课程信息表中删除指定课程编号的记录。（见教材第222页）

- **4. 参数化查询**
- **如果要创建一个使用多次但每次使用不同值的查询，那么应在查询中使用参数，即创建参数查询。参数查询避免了在每次查询中重新建立SQL查询语句,并且可以减少SQL注入漏洞的概率，增强数据库应用程序的安全。**

- 例9.4 查询教师信息表“Instructor”中指定教师所教授的课程，并显示查询结果。（见教材第223页）

- **9.2.3 DataReader对象的使用**
- **DataReader对象可以看作是一个简单的数据集，其主要用于从数据源中查询数据。DataReader类被设计为产生只读和只进的数据流，因而只能对数据源进行逐行访问，即每次只能有一行记录保存在服务器的内存中。**

- 1. 创建DataReader对象
- 使用DataReader的时候，不能直接通过构造函数实例化DataReader类，通常我们使用执行Command类的ExecuteRader方法来返回DataReader对象（如例9.1所示），语法结构如下：
- Dim DataReader对象as OleDbDataReader =Command对象.ExecuteReader()

- **2. 使用DataReader对象读取数据**
- **DataReader类最常见的用法就是SQL查询或者存储过程返回的记录集。在使用它时，数据库连接必须保持打开状态，另外只能从前往后遍历信息，不能中途停下修改数据。**
- **1）遍历DataReader中的记录**
- **DataReader对象的Read方法可以判断DataReader对象中的数据是否还有下一行，并将游标下移到下一行，通过该方法可以遍历读取数据库中行的信息，示例代码如下：**
- **Dim objCmd as New OleDbCommand（”Select * from Instructor”,objConn）**
- **Dim objReader as OleDbDataReader=objCmd.ExecuteReader()**
- **while(objReader.Reader())**
- **{**
- **‘逐行处理记录**
- **}**

- **2）DataReader访问字段的值**
- **从DataReader对象获取数据有两种方法：第一种是Item属性，此属性返回字段索引或者字段名字对应的字段的值。第二种是Get方法，此方法返回由字段索引指定的字段的值。**
- **假如现有一个objReader实例对应的SQL语句是“select InstruName, Sex from Instructor”，则可以使用下面的示例代码取得字段的值。**
- **‘使用Item属性**
- **Dim name as String=objReader(“Instruname”) ‘使用字段名**
- **Dim sex as String=objReader(“sex”) ‘使用字段名**
- **或者：**
- **Dim name as String=objReader(0) ‘使用字段索引**
- **Dim sex as String=objReader(1) ‘使用字段索引**
- **‘使用Get方法**
- **Dim name as String = objReader.GetString(0)**
- **Dim sex as String = objReader.GetString(1)**